**GREATEST OF THREE NUMBER :**

#include<iostream.h>

#include<conio.h>

class compare

{

int a,c,b;

public:

void getdata()

{

cout<<"Enter the three number "<<endl;

cin>>a>>b>>c;

}

void showdata()

{

if(a>b&&a>c)

cout<<a<<"is the greatest";

else if(b>a && b>c)

cout<<b<<"is the greatest";

else

cout<<c<<"is the greatest";

} };

void main()

{

clrscr();

compare s;

s.getdata();

s.showdata();

getch();

}

**OUTPUT :**

Enter the three number 1 7 10

10 is the greatest

**PRIME NUMBER GENERATION :**

#include<iostream.h>

#include<conio.h>

class prime

{

int i,n,count,m,flag;

public:

void execute()

{

cout<<"Enter the limit of prime number"<<endl;

cin>>n;

for(i=1;i<=n;i++)

{

flag=0;

for(i=2;i<n/2;i++)

{

if(n%i==0)

{

flag=1;

break;

}

if(flag!=1)

{

cout<<m<<endl;

}

}

} } };

void main()

{

clrscr();

prime p;

p.execute();

getch();

}

**OUTPUT:**

Enter the limit of prime number 5

2

3

5

7

11

**STRING LENGTH WITHOUT LIBRARY FUNCTION :**

#include<iostream.h>

#include<conio.h>

class length

{

int length,i;

char s[10];

public:

void getdata()

{

cout<<"Enter the string"<<endl;

cin>>s;

}

void cal()

{

for(i=0;s[i]!='\0';i++)

{

length++;

}

cout<<"length of string is "<<length; }};

void main()

{

clrscr();

length l;

l.getdata();

l.cal();

getch();

}

**OUTPUT :**

Enter the string

Doctor

length of string is 6

**FACTORIAL OF A NUMBER USING RECURSIVE FUNCTION :**

#include <iostream>

using namespace std;

int factorial(int);

int main() {

int n;

cout<<"Enter a number to find factorial: ";

cin>>n;

cout<<"Factorial of "<<n<<" = "<<factorial(n);

return 0;

}

int factorial(int n)

{

if (n>1) {

return n\*factorial(n-1);

}

else {

return 1;

}

}

OUTPUT :

Enter a number to find factorial 5

Factorial of 5 is 25

**COMPUTING AREA AND CIRCUMFERENCE OF CIRCLE WITH CLASS :**

#include<iostream.h>

#include<conio.h>

class calculation{

float area,cirum,a;

public:

void getdata()

{

cout<<"Enter the radius of circle "<<endl;

cin>>a;

}

void area()

{

area=3.14\*a\*a;

cout<<"Area is "<<area<<endl;

}

void circumference()

{

cirum=2.14\*3.14\*a;

cout<<"Circumference "<<cirum;

} };

void main()

{

clrscr();

calculation s;

s.getdata();

s.area();

s.circumference();

getch();

}

**OUTPUT :**

Enter the radius of circle 4

Area is 50.24

Circumference 24.8

**BANK ACCOUNT USING ARRAY OF OBJECT :**

#include<iostream.h>

#include<conio.h>

class detail

{

int acc,otp;

char s[10];

public:

void getdata()

{

cout<<endl;

cout<<"Enter your account number"<<endl;

cin>>acc;

cout<<endl;

cout<<"Enter your name "<<endl;

cin>>s;

cout<<endl;

cout<<"Enter your otp number"<<endl;

cin>>otp;

}

void showdata()

{

cout<<endl;

cout<<"Your account number is "<<acc<<endl;

cout<<"Your name number is "<<s<<endl;

}

};

void main()

{

clrscr();

detail s[5];

for(int i=0;i<5;i++)

{

s[i].getdata();

}

for(i=0;i<5;i++)

{

cout<<endl;

s[i].showdata();

}

getch(); }

**OUTPUT :**

Enter your account number 14107

Enter your name Raj

Enter your otp number 42457

Your account number is 14107

Your name is Raj

**COPY CONSTRUCTOR :**

#include<conio.h>

class copy

{

int rollno;

public:

copy()

{

rollno=47;

}

copy(copy &obj)

{

rollno=obj.rollno;

}

void display()

{

cout<<"The rollno is "<<" "<<rollno<<endl;

}

};

void main()

{

clrscr();

copy c1;

c1.display();

cout<<"calling copy constructor"<<endl;

copy c2(c1);

c2.display();

getch();

}

OUTPUT :

The rollno is 47

calling copy constructor

The rollno is 47

**DYNAMIC CONSTRUCTOR :**

#include<iostream.h>

#include<conio.h>

class dynamic

{

int \*ptr;

public:

dynamic(int val)

{

ptr=new int;

\*ptr =val;

}

~dynamic()

{

cout<<"Deleting the address";

delete ptr;

}

void showdata()

{

cout<<"Value of received data"<<\*ptr;

}

};

int main()

{

int no1;

clrscr();

cout<<"Enter the number"<<endl;

cin>>no1;

dynamic s1(no1);

s1.showdata();

getch();

return 0;

}

OUTPUT :

Enter the number 7

Value of received data 7

**STACK IMPLEMENTATION USING ARRAY :**

#include<iostream.h>

#include<conio.h>

#define max 5

class stack

{

int stk[max],top,element;

public:

stack();

int isfull();

int isempty();

void push(int);

void pop();

void peep();

};

stack::stack()

{top=-1;

}

int stack::isfull()

{if (top==max-1)

return 1;

return 0;

}

int stack::isempty()

{

if(top==-1)

return 1;

return 0;

}

void stack::push(int ele)

{

if(!isfull())

{

top++;

stk[top]=ele;

}

else

{

cout <<"stack is full";

} }

void stack::pop()

{

if(!isempty())

{

cout<<stk[top]<<"deleted"<<endl;

top--;

}

else

{

cout<<"stack is empty";

}

}

void stack::peep()

{int i;

if(! isempty())

{

for(i=0;i<=top;i++)

cout<<stk[i];

}

else

cout<<"empty stack"<<endl;

}

void main()

{

clrscr();

stack s;

int choice,element;

char a;

do

{

cout<<" Enter 1 for push , 2 for pop and 3 for peep function call "<<endl;

cin>>choice;

switch(choice)

{

case 1:

cout<<"enter the element to push";

cin>>element;

s.push(element);

break;

case 2:

s.pop();

break;

case 3:

s.peep();

break;

default:

cout<<"invalid choice"<<endl;

}

cout<<" You do want to continue ? y/n";

cin>>a;

}

while(a!='n');

getch();

}

**OUTPUT :**

Enter 1 for push , 2 for pop and 3 for peep function call

1

Enter the element 5

You do want to continue ? y/n

y

Enter 1 for push , 2 for pop and 3 for peep function call

1

Enter the element 7

You do want to continue ? y/n

y

Enter 1 for push , 2 for pop and 3 for peep function call

3

The following are the element

7

5

**STACK IMPLEMENTATION USING POINTER :**

#include<conio.h>

struct node

{

int data;

struct node \*next;

};

class stack

{

struct node \*top;

public:

stack()

{

top=NULL;

}

void push();

void peep();

void pop();

};

void stack::push()

{

node \*temp;

temp=new node;

cout<<"Enter the number "<<endl;

cin>>temp->data;

temp->next=NULL;

if(top==NULL)

top=temp;

else

{

temp->next=top;

top=temp;

}

}

void stack::pop()

{

node \*temp;

if(top!=NULL)

{

temp=top;

cout<<temp->data<<endl;

top=top->next;

delete temp;

}

else

{

delete temp->next;

cout<<"Nothing to pop ";

}

}

void stack::peep()

{

node \*temp;

for(temp=top;temp!=NULL;temp=temp->next)

{

cout<<temp->data<<endl;

}

}

void main()

{

clrscr();

int choice;

char ans;

stack s;

do

{

cout<<"Enter 1 to push 2 to pop 3 to peep "<<endl<<endl;

cin>>choice;

switch(choice)

{

case 1:

{

s.push();

break;

}

case 2:

{ s.pop();

break;

}

case 3:

{

s.peep();

break;

} default :cout<<"invalid number";

}

cout<<"Enter y/n to contiune "<<endl<<endl;

cin>>ans;

}while(ans!='n');

getch();

}

OUTPUT :

Enter 1 for push , 2 for pop and 3 for peep function call

1

Enter the element 5

You do want to continue ? y/n

y

Enter 1 for push , 2 for pop and 3 for peep function call

1

Enter the element 7

You do want to continue ? y/n

y

Enter 1 for push , 2 for pop and 3 for peep function call

3

The following are the element

7

5

**QUEUE IMPLEMETATION USING ARRAY :**

#include<iostream.h>

#include<conio.h>

#define max 5

class queue

{

private:

int rear,front,q[max];

public:

queue();

void enqueue(int);

void dequeue();

void print();

};

queue::queue()

{

rear=-1;

front=0;

}

void queue::enqueue(int x)

{

if(rear<=max-1)

{

rear++;

q[rear]=x;

}

else

{

cout<<"queue is full";

}

}

void queue::dequeue()

{

if(rear>=front)

{

cout<<q[front];

front++;

}

else

{

cout<<"queue is empty"; }}

void queue::print()

{

if(rear>=front)

{

for(int i=front;i<=rear;i++)

cout<<q[i];

}

else

{

cout<<"queue is empty";

}

}

void main()

{

clrscr();

queue s;

int choice,element;

char a;

do

{

cout<< " Enter 1 for enqueue, 2 for dequeue and 3 for print "<<endl;

cin>>choice;

switch(choice)

{

case 1:

cout<<"\nenter the element to enqueue";

cin>>element;

s.enqueue(element);

break;

case 2:

s.dequeue();

break;

case 3:

s.print();

break;

default:

cout<<"invalid choice"<<endl;

}

cout<<"You do want to continue ? y/n "<<endl;

cin>>a;

} while(a!='n');

getch();

}

OUTPUT :

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 5

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 7

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

3

The following are the element

5

7

**QUEUE USING POINTER** :

#include<iostream.h>

#include<conio.h>

struct node

{

int val;

struct node \*next;

};

class qptr

{

struct node \*front;

struct node \*rear;

public:

qptr();

void enqueue(int);

void dequeue();

void print();

};

qptr::qptr()

{

front=NULL;

rear=NULL;

}

void qptr::enqueue(int x)

{

struct node \*temp;

temp=new node;

temp->val=x;

temp->next=NULL;

if((rear==NULL)&&(front==NULL))

{

front=rear=temp;

}

else

{

rear->next=temp;

rear=temp;

}

}

void qptr::dequeue()

{

if((front==NULL)&&(rear==NULL))

{

cout<<"queue is empty";

}

else

{

cout<<front->val<<"deleted";

front=front->next;

}

if(front==NULL)

{

rear=NULL;

}

}

void qptr::print()

{

struct node \*temp;

if((front!=NULL)&&(rear!=NULL))

{

for(temp=front;temp!=NULL;temp=temp->next)

{

cout<<temp->val;

}

}

else

{

cout<<"queue is empty";

}

}

void main()

{

clrscr();

qptr s;

int choice,element;

char a;

do

{

cout <<"Enter 1 for enqueue, 2 for dequeue and 3 for print "<<endl;

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the element to enqueue";

cin>>element;

s.enqueue(element);

break;

case 2:

s.dequeue();

break;

case 3:

s.print();

break;

default:

cout<<"invalid choice"<<endl;

}

cout<<" You do want to continue ? y/n ";

cin>>a;

} while(a!='n');

getch();

}

OUTPUT :

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 5

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 7

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

3

The following are the element

5

7

DECIMAL TO BINARY USING STACK IN PROXY CLASS

'stroxy.h'

#include<iostream.h>

#include<conio.h>

#define max 10

class stack

{

private:

int top,stk[max];

public:

stack();

int isfull();

int isempty();

void push(int);

void pop();

void peep();

};

#include<iostream.h>

#include"c:\TCC\stroxy.H"

stack::stack()

{

top=-1;

}

int stack::isfull()

{

if(top==max-1)

return 1;

return 0;

}

int stack::isempty()

{

if(top==-1)

return 1;

return 0;

}

void stack::push(int ele)

{

if(!isfull())

{

top++;

stk[top]=ele;

}

else

{

cout<<"stack is full";

}

}

void stack::pop()

{

if(!isempty())

{

cout<<stk[top]<<"deleted"<<endl;

top--;

}

else

{

cout<<"stack is empty";

}

}

void stack::peep()

{

int i;

if(!isempty())

{

for(i=top;i>=0;i--)

{

cout<<stk[i];

}

}

else

{

cout<<"empty stack"<<endl;

}

}

#include<iostream.h>

#include"c:\TCC\stroxy1.cpp"

void main()

{

clrscr();

stack s;

int n,r;

cout<<"Enter the decimal no:";

cin>>n;

while(n>0)

{

r=n%2;

s.push(r);

n=n/2;

}

cout<<" The binary is";

s.peep();

getch();

}

OUTPUT :

Enter the decimal number 7

The binary is 0111

CURSOR IMPLEMENTATION USING STACK :

**“curslist.h” File:**  
  
#include<stdio.h>  
#include<conio.h>  
#define SPACE\_SIZE 10  
struct Node  
{  
   int data;  
   int Next;  
};  
typedef int PtrToNode;  
typedef PtrToNode POSITION;  
typedef PtrToNode LIST;  
struct Node cursor[SPACE\_SIZE];  
  
void InitializeCursor()  
{  
   int i;  
   for(i=0;i<=SPACE\_SIZE-1;i++)  
   {  
      cursor[i].Next=i+1;  
      cursor[i].data=0;  
   }  
      cursor[SPACE\_SIZE-1].Next=-1;  
}  
POSITION CursorAlloc()  
{  
   POSITION P;  
   P=cursor[0].Next;  
   cursor[0].Next=cursor[P].Next;  
   cursor[P].data=-1;  
   cursor[P].Next=-1;  
   return P;  
}  
void CursorFree(POSITION P)  
{  
   cursor[P].Next=cursor[0].Next;  
   cursor[0].Next=P;  
   cursor[P].data=0;  
}  
void Insert(int X,POSITION P)  
{  
      POSITION Temp;  
      Temp=CursorAlloc();  
      if(Temp==-1)  
     printf("\nOut of space");  
      else if(cursor[P].data==0)  
     printf("\nPosition is not in the list");  
      else  
      {  
  
     cursor[Temp].data=X;  
     cursor[Temp].Next=cursor[P].Next;  
     cursor[P].Next=Temp;  
      }  
   }  
int IsLast(POSITION P)  
{  
   return cursor[P].Next==-1;  
}  
int IsEmpty(LIST L)  
{  
   return cursor[L].Next==-1;  
}  
POSITION Find(int X,LIST L)  
{  
   POSITION Temp;  
   Temp=cursor[L].Next;  
   while(Temp!=-1&&cursor[Temp].data!=X)  
   Temp=cursor[Temp].Next;  
   return Temp;  
}  
POSITION FindPrevious(int X,LIST L)  
{  
   POSITION Temp;  
   Temp=L;  
   while(Temp!=-1&&cursor[cursor[Temp].Next].data!=X)  
   Temp=cursor[Temp].Next;  
   return Temp;  
}  
void Delete(int X,LIST L)  
{  
   POSITION P,Temp;  
   P=FindPrevious(X,L);  
   if(!IsLast(P))  
   {  
      Temp=cursor[P].Next;  
      cursor[P].Next=cursor[Temp].Next;  
      CursorFree(Temp);  
   }  
}  
void MakeEmpty(LIST L)  
{  
   while(!IsEmpty(L))  
   Delete(cursor[cursor[L].Next].data,L);  
}  
void Display()  
{  
   int i;  
   for(i=0;i<=SPACE\_SIZE-1;i++)  
   printf("\n%d\t%d\t%d",i,cursor[i].data,cursor[i].Next);  
}  
  
  
**“curslist.c” File:**  
  
#include<stdio.h>  
#include<conio.h>  
#include"curslist.h"  
void main()  
{  
LIST L=-1;  
POSITION P;  
int choice,place,x;  
clrscr();  
printf("\n1.Create\n2.Insert\n3.Delete\n4.MakeEmpty\n5.Display\n6.Find\n7.Exit");  
A:  
printf("\nEnter ur choice:\t");  
scanf("%d",&choice);  
switch(choice)  
{  
case 1:  
    if(L==-1)  
    {  
       InitializeCursor();  
       L=CursorAlloc();  
    }  
    else  
       printf("\nList is already created");  
    break;  
case 2:  
    if(L==-1)  
       printf("\nList is not yet initialized");  
    else  
    {  
       printf("\nWhere u want to insert?");  
       scanf("%d",&place);  
       printf("\nEnter the element to insert");  
       scanf("%d",&x);  
       Insert(x,place);  
    }  
    break;  
case 3:  
    if(L==-1)  
       printf("\nList is not yet initialized");  
    else  
    {  
       printf("\nWhich element you want to delete?");  
       scanf("%d",&x);  
       Delete(x,L);  
    }  
    break;  
case 4:  
    if(L==-1)  
       printf("\nList is not yet initialized");  
    else  
       MakeEmpty(L);  
    break;  
case 5:  
    if(L==-1)  
       printf("\nList is not yet initialized");  
    else  
       Display();  
    break;  
case 6:  
    if(L==-1)  
       printf("\nList is not yet initialized");  
    else  
    {  
       printf("\nWhich element you want to search?");  
       scanf("%d",&x);  
       P=Find(x,L);  
       printf("\nThe element is at %d",P);  
    }  
    break;  
case 7:  
    exit(0);  
default:  
    printf("\n \*\*\*\*\*\*\*WRONG ENTRY\*\*\*\*\*\*\*");  
}  
goto A;  
}  
  
**OUTPUT:**  
  
1.Create  
2.Insert  
3.Delete  
4.MakeEmpty  
5.Display  
6.Find  
7.Exit  
Enter ur choice:    1  
  
Enter ur choice:    5  
0       0       2  
1       -1      -1  
2       0       3  
3       0       4  
4       0       5  
5       0       6  
6       0       7  
7       0       8  
8       0       9  
9       0       -1  
  
Enter ur choice:    2  
Where u want to insert?    1  
Enter the element to insert:    100  
  
Enter ur choice:        5  
0       0       3  
1       -1      2  
2       100     -1  
3       0       4  
4       0       5  
5       0       6  
6       0       7  
7       0       8  
8       0       9  
9       0       -1

CIRCULAR QUEUE :

#include<iostream.h>

#include<conio.h>

#define max 5

class cirq

{

int front,rear,q[max];

public:

cirq();

void enqueue(int);

void dequeue();

void print();

int isfull();

int isempty();

};

cirq::cirq()

{

front=1;

rear=max;

}

int cirq::isfull()

{

return((rear+2)%max==front);

}

int cirq::isempty()

{

return((rear+1)%max==front);

}

void cirq::enqueue(int x)

{

if(!isfull())

{

rear=(rear+1)%max;

q[rear]=x;

}

else

{

cout<<"queue is full";

}

}

void cirq::dequeue()

{

if(!isempty())

{

cout<<q[front]<<"deleted";

front=(front+1)%max;

}

else

{

cout<<"queue is empty";

}

}

void cirq::print()

{

if(!isempty())

{

for(int i=front;i!=rear;i=(i+1)%max)

cout<<q[i];

cout<<q[rear];

}

else

cout<<"queue is empty"; }

void main()

{

clrscr();

cirq s;

int choice,element;

char a;

do

{

cout<<" Enter 1 for enqueue, 2 for dequeue and 3 for print "<<endl;

cin>>choice;

switch(choice)

{

case 1:

{

cout<<"\nenter the element to enqueue";

cin>>element;

s.enqueue(element);

break;

}

case 2:

s.dequeue();

break;

case 3:

s.print();

break;

default:

cout<<"invalid choice"<<endl;

}

cout<<" You do want to continue ? y/n ";

cin>>a;

}

while(a!='n');

getch();

}

**OUTPUT :**

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 5

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

1

Enter the element 7

You do want to continue ? y/n

y

Enter 1 for enqueue, 2 for dequeue and 3 for print

3

The following are the element

5

7

LINKED LIST USING ARRAY :

#include<iostream.h>

#include<conio.h>

#define max 5

class sll

{

int last,list[max];

public:

sll();

void insertbeg();

void insertend();

void insertmid();

void deletebeg();

void deleteend();

void deletemid();

void traverse();

};

sll::sll()

{

int i;

last=0;

for(i=0;i<max;i++)

list[i]=-1;

}

void sll::insertbeg()

{

int i,value;

cout<<"Enter the value";

cin>>value;

last++;

for(i=last-1;i>0;i--)

list[i]=list[i-1];

list[0]=value;

}

void sll::insertend()

{

int value;

cout<<"enter the value";

cin>>value;

last++;

list[last-1]=value;

}

void sll::insertmid()

{

int i,value,pos;

cout<<"position";

cin>>pos;

if(pos==1)

{

insertbeg();

return;

}

if(pos==last)

{

insertend();

return; }

if((pos<1)||(pos>last))

{

cout<<"invalid position";

}

else

{

cout<<"value";

cin>>value;

last++;

for(i=last-1;i>=pos;i--)

list[i]=list[i-1];

list[pos-1]=value;

} }   
void sll::deletebeg()

{

int i,value;

if(last==0)

return;

cout<<list[0]<<"deleted";

for(i=1;i<last;i++)

list[i-1]=list[i];

last--; }

void sll::deleteend()

{

int i;

if(last==0)

return;

cout<<list[last-1]<<"deleted";

last--;

}

void sll::deletemid()

{

int i,pos;

cout<<"position";

cin>>pos;

if(pos==1)

{

deletebeg();

return;

}

if(pos==last)

{

deleteend();

return; }

if((pos<1)||(pos>last))

{

cout<<"invalid position";

return;

}

else

{

cout<<list[pos-1]<<"deleted";

for(i=pos;i<last;i++)

list[i-1]=list[i];

last--;

}

}

void sll::traverse()

{

for(int i=0;i<last;i++)

{

cout<<list[i];

}

}

void main()

{

clrscr();

sll s;

int choice;

char a;

do

{

cout<<"Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op"<<endl;

cin>>choice;

switch(choice)

{

case 1:

s.insertbeg();

break;

case 2:

s.insertend();

break;

case 3:

s.insertmid();

break;

case 4:

s.deletebeg();

break;

case 5:

s.deleteend();

break;

case 6:

s.deletemid();

break;

case 7:

s.traverse();

break;

default:

cout<<"invlaid choice";

}

cout<<" You do want to continue ? y/n ";

cin>>a;

}while(a!='n');

getch();

}

OUTPUT :

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

1

Enter the element 5

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

1

Enter the element 7

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

2

Enter the element 8

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

4

Enter the element 9

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

5

7

8

9

LINKED LIST USING POINTER :

#include<iostream.h>

#include<conio.h>

struct node

{

int data;

struct node \*next;

};

class listadt

{

struct node \*head;

public:

listadt();

void insertbeg();

void insertend();

void insertmid();

void deletebeg();

void deleteend();

void deletemid();

void traverse();

};

listadt::listadt()

{

head=NULL;

}

void listadt::insertbeg()

{

struct node \*temp;

temp=new node;

cout<<"get value";

cin>>temp->data;

temp->next=NULL;

if(temp==NULL)

{

head=temp;

}

else

{

temp->next=head;

head=temp;

}

}

void listadt::insertend()

{

struct node \*temp,\*curr;

temp=new node;

cout<<"get value";

cin>>temp->data;

temp->next=NULL;

if(temp==NULL)

{

head=temp;

}

else

for(curr=head;curr->next!=NULL;curr=curr->next);

curr->next=temp;

}

void listadt::insertmid()

{

struct node \*temp,\*curr,\*prev;

int i,pos;

cout<<"enter the position"<<endl;

cin>>pos;

if(pos==1)

{

insertbeg();

return;

}

for(i=1,prev=head,curr=head;i!=pos&&curr->next!=NULL;i++,prev=curr,curr=curr->next);

if(i==pos)

{

temp=new node;

cout<<"enter the no";

cin>>temp->data;

temp->next=NULL;

temp->next=curr;

prev->next=temp;

}

else

{

cout<<"invalid";

}

}

void listadt::deletebeg()

{

struct node \*temp;

if(head!=NULL)

{

temp=head;

cout<<temp->data<<"deleted";

head=head->next;

delete temp;

}

else

cout<<"list is empty";

}

void listadt::deleteend()

{

struct node \*prev,\*curr;

if(head!=NULL)

{

for(prev=head,curr=head;curr->next!=NULL;prev=curr,curr=curr->next);

{

cout<<curr->data<<"deleted";

if(prev!=curr)

{

prev->next=NULL;

delete curr;

}

else

head=NULL;

}

}

else

cout<<"empty";

}

void listadt::deletemid()

{

struct node \*prev,\*curr;

int i,pos;

cout<<"enter the position"<<endl;

cin>>pos;

if(head==NULL)

{

return;

}

if(pos==1)

{

deletebeg();

return;

}

for(i=1,prev=head,curr=head;i!=pos&&curr->next!=NULL;i++,prev=curr,curr=curr->next);

if(pos==i)

{

cout<<curr->data<<"delete";

if(prev->next==curr->next)

delete curr;

}

else

cout<<"invalid pos";

}

void listadt::traverse()

{

struct node \*temp;

if(head!=NULL)

{

cout<<"list is as follows\n";

for(temp=head;temp!=NULL;temp=temp->next)

cout<<temp->data<<endl;

}

else

cout<<"list empty\n";

}

void main()

{

clrscr();

listadt s;

int n,choice;

do

{

cout<<"enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op";

cin>>choice;

switch(choice)

{

case 1:

s.insertbeg();

break;

case 2:

s.insertend();

break;

case 3:

s.insertmid();

break;

case 4:

s.deletebeg();

break;

case 5:

s.deleteend();

break;

case 6:

s.deletemid();

break;

case 7:

s.traverse();

break;

default:

cout<<"invlaid choice";

}

cout<<" You do want to continue ? y/n ";

cin>>a;

}while(a!='n');

getch();

}

OUTPUT :

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

1

Enter the element 5

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

1

Enter the element 7

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

2

Enter the element 8

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

4

Enter the element 9

You do want to continue ? y/n

y

Enter choice 1.ib 2.ie 3.im 4.db 5.de 6.dm 7.op

5

7

8

9

DOUBLE LINKED LIST :

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int info;

struct node \*next;

struct node \*prev;

}\*start;

class double\_llist

{

public:

void create\_list(int value);

void add\_begin(int value);

void add\_after(int value, int position);

void delete\_element(int value);

void search\_element(int value);

void display\_dlist();

void count();

void reverse();

double\_llist()

{

start = NULL;

}

};

void double\_llist::create\_list(int value)

{

struct node \*s, \*temp;

temp = new(struct node);

temp->info = value;

temp->next = NULL;

if (start == NULL)

{

temp->prev = NULL;

start = temp;

}

else

{

s = start;

while (s->next != NULL)

s = s->next;

s->next = temp;

temp->prev = s;

}

}

void double\_llist::add\_begin(int value)

{

if (start == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->prev = NULL;

temp->info = value;

temp->next = start;

start->prev = temp;

start = temp;

cout<<"Element Inserted"<<endl;

}

void double\_llist::add\_after(int value, int pos)

{

if (start == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*tmp, \*q;

int i;

q = start;

for (i = 0;i < pos - 1;i++)

{

q = q->next;

if (q == NULL)

{

cout<<"There are less than ";

cout<<pos<<" elements."<<endl;

return;

}

}

tmp = new(struct node);

tmp->info = value;

if (q->next == NULL)

{

q->next = tmp;

tmp->next = NULL;

tmp->prev = q;

}

else

{

tmp->next = q->next;

tmp->next->prev = tmp;

q->next = tmp;

tmp->prev = q;

}

cout<<"Element Inserted"<<endl;

}

void double\_llist::delete\_element(int value)

{

struct node \*tmp, \*q;

if (start->info == value)

{

tmp = start;

start = start->next;

start->prev = NULL;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = start;

while (q->next->next != NULL)

{

if (q->next->info == value)

{

tmp = q->next;

q->next = tmp->next;

tmp->next->prev = q;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = q->next;

}

if (q->next->info == value)

{

tmp = q->next;

free(tmp);

q->next = NULL;

cout<<"Element Deleted"<<endl;

return;

}

cout<<"Element "<<value<<" not found"<<endl;

}

void double\_llist::display\_dlist()

{

struct node \*q;

if (start == NULL)

{

cout<<"List empty,nothing to display"<<endl;

return;

}

q = start;

cout<<"The Doubly Link List is :"<<endl;

while (q != NULL)

{

cout<<q->info<<" <-> ";

q = q->next;

}

cout<<"NULL"<<endl;

}

void double\_llist::count()

{

struct node \*q = start;

int cnt = 0;

while (q != NULL)

{

q = q->next;

cnt++;

}

cout<<"Number of elements are: "<<cnt<<endl;

}

void double\_llist::reverse()

{

struct node \*p1, \*p2;

p1 = start;

p2 = p1->next;

p1->next = NULL;

p1->prev = p2;

while (p2 != NULL)

{

p2->prev = p2->next;

p2->next = p1;

p1 = p2;

p2 = p2->prev;

}

start = p1;

cout<<"List Reversed"<<endl;

}

int main()

{

int choice, element, position;

double\_llist dl;

while (1)

{

cout<<endl<<"----------------------------"<<endl;

cout<<endl<<"Operations on Doubly linked list"<<endl;

cout<<endl<<"----------------------------"<<endl;

cout<<"1.Create Node"<<endl;

cout<<"2.Add at begining"<<endl;

cout<<"3.Add after position"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Display"<<endl;

cout<<"6.Count"<<endl;

cout<<"7.Reverse"<<endl;

cout<<"8.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch ( choice )

{

case 1:

cout<<"Enter the element: ";

cin>>element;

dl.create\_list(element);

cout<<endl;

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

dl.add\_begin(element);

cout<<endl;

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert Element after postion: ";

cin>>position;

dl.add\_after(element, position);

cout<<endl;

break;

case 4:

if (start == NULL)

{

cout<<"List empty,nothing to delete"<<endl;

break;

}

cout<<"Enter the element for deletion: ";

cin>>element;

dl.delete\_element(element);

cout<<endl;

break;

case 5:

dl.display\_dlist();

cout<<endl;

break;

case 6:

dl.count();

break;

case 7:

if (start == NULL)

{

cout<<"List empty,nothing to reverse"<<endl;

break;

}

dl.reverse();

cout<<endl;

break;

case 8:

exit(1);

default:

cout<<"Wrong choice"<<endl;

}

}

return 0;

}

OUTPUT :

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Count

7.Reverse

8.Quit

Enter your choice : 2

Enter the element: 100

First Create the list.

---------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Count

7.Reverse

8.Quit

Enter your choice : 3

Enter the element: 200

Insert Element after postion: 1

First Create the list.

--------------------------------

Operations on Doubly linked list

---------------------------------

1.Create Node

2.Add at begining

3.Add after

4.Delete

5.Display

6.Count

7.Reverse

8.Quit

Enter your choice : 4

List empty,nothing to delete

CIRCULAR LINKED LIST :

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int info;

struct node \*next;

}\*last;

class circular\_llist

{

public:

void create\_node(int value);

void add\_begin(int value);

void add\_after(int value, int position);

void delete\_element(int value);

void search\_element(int value);

void display\_list();

void update();

void sort();

circular\_llist()

{

last = NULL;

}

};

void circular\_llist::create\_node(int value)

{

struct node \*temp;

temp = new(struct node);

temp->info = value;

if (last == NULL)

{

last = temp;

temp->next = last;

}

else

{

temp->next = last->next;

last->next = temp;

last = temp;

}

}

void circular\_llist::add\_begin(int value)

{

if (last == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->info = value;

temp->next = last->next;

last->next = temp;

}

void circular\_llist::add\_after(int value, int pos)

{

if (last == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp, \*s;

s = last->next;

for (int i = 0;i < pos-1;i++)

{

s = s->next;

if (s == last->next)

{

cout<<"There are less than ";

cout<<pos<<" in the list"<<endl;

return;

}

}

temp = new(struct node);

temp->next = s->next;

temp->info = value;

s->next = temp;

if (s == last)

{

last=temp;

}

}

void circular\_llist::delete\_element(int value)

{

struct node \*temp, \*s;

s = last->next;

if (last->next == last && last->info == value)

{

temp = last;

last = NULL;

free(temp);

return;

}

if (s->info == value) /\*First Element Deletion\*/

{

temp = s;

last->next = s->next;

free(temp);

return;

}

while (s->next != last)

{

if (s->next->info == value)

{

temp = s->next;

s->next = temp->next;

free(temp);

cout<<"Element "<<value;

cout<<" deleted from the list"<<endl;

return;

}

s = s->next;

}

if (s->next->info == value)

{

temp = s->next;

s->next = last->next;

free(temp);

last = s;

return;

}

cout<<"Element "<<value<<" not found in the list"<<endl;

}

void circular\_llist::search\_element(int value)

{

struct node \*s;

int counter = 0;

s = last->next;

while (s != last)

{

counter++;

if (s->info == value)

{

cout<<"Element "<<value;

cout<<" found at position "<<counter<<endl;

return;

}

s = s->next;

}

if (s->info == value)

{

counter++;

cout<<"Element "<<value;

cout<<" found at position "<<counter<<endl;

return;

}

cout<<"Element "<<value<<" not found in the list"<<endl;

}

void circular\_llist::display\_list()

{

struct node \*s;

if (last == NULL)

{

cout<<"List is empty, nothing to display"<<endl;

return;

}

s = last->next;

cout<<"Circular Link List: "<<endl;

while (s != last)

{

cout<<s->info<<"->";

s = s->next;

}

cout<<s->info<<endl;

}

void circular\_llist::update()

{

int value, pos, i;

if (last == NULL)

{

cout<<"List is empty, nothing to update"<<endl;

return;

}

cout<<"Enter the node position to be updated: ";

cin>>pos;

cout<<"Enter the new value: ";

cin>>value;

struct node \*s;

s = last->next;

for (i = 0;i < pos - 1;i++)

{

if (s == last)

{

cout<<"There are less than "<<pos<<" elements.";

cout<<endl;

return;

}

s = s->next;

}

s->info = value;

cout<<"Node Updated"<<endl;

}

void circular\_llist::sort()

{

struct node \*s, \*ptr;

int temp;

if (last == NULL)

{

cout<<"List is empty, nothing to sort"<<endl;

return;

}

s = last->next;

while (s != last)

{

ptr = s->next;

while (ptr != last->next)

{

if (ptr != last->next)

{

if (s->info > ptr->info)

{

temp = s->info;

s->info = ptr->info;

ptr->info = temp;

}

}

else

{

break;

}

ptr = ptr->next;

}

s = s->next;

}

}

int main()

{

int choice, element, position;

circular\_llist cl;

while (1)

{

cout<<endl<<"---------------------------"<<endl;

cout<<endl<<"Circular singly linked list"<<endl;

cout<<endl<<"---------------------------"<<endl;

cout<<"1.Create Node"<<endl;

cout<<"2.Add at beginning"<<endl;

cout<<"3.Add after"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Search"<<endl;

cout<<"6.Display"<<endl;

cout<<"7.Update"<<endl;

cout<<"8.Sort"<<endl;

cout<<"9.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the element: ";

cin>>element;

cl.create\_node(element);

cout<<endl;

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

cl.add\_begin(element);

cout<<endl;

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert element after position: ";

cin>>position;

cl.add\_after(element, position);

cout<<endl;

break;

case 4:

if (last == NULL)

{

cout<<"List is empty, nothing to delete"<<endl;

break;

}

cout<<"Enter the element for deletion: ";

cin>>element;

cl.delete\_element(element);

cout<<endl;

break;

case 5:

if (last == NULL)

{

cout<<"List Empty!! Can't search"<<endl;

break;

}

cout<<"Enter the element to be searched: ";

cin>>element;

cl.search\_element(element);

cout<<endl;

break;

case 6:

cl.display\_list();

break;

case 7:

cl.update();

break;

case 8:

cl.sort();

break;

case 9:

exit(1);

break;

default:

cout<<"Wrong choice"<<endl;

}

}

return 0;

}

OUTPUT :

---------------------------------

Operations on Circular singly linked list

---------------------------------

1.Create Node

2.Add at beginning

3.Add after

4.Delete

5.Search

6.Display

7.Update

8.Sort

9.Quit

Enter your choice : 6

List is empty, nothing to display

---------------------------------

Operations on Circular singly linked list

---------------------------------

1.Create Node

2.Add at beginning

3.Add after

4.Delete

5.Search

6.Display

7.Update

8.Sort

9.Quit

Enter your choice : 7

List is empty, nothing to update